REMARKS

Claims 1-2, 22, 30-32, 45-47, 58, 63 and 74-75 have been amended as suggested by the Examiner for non-substantive informalities. No claims have been canceled or added as part of this Reply. Accordingly, claims 1-85 are currently pending in the instant patent application.

A. Claim Objections

The Examiner has objected to claims 1-2, 22, 30-32, 45-47, 58, 63 and 74-75 because each contained more than one capital letter. Amendments have been made as requested by the Examiner to correct this problem.

B. Claim Rejections

In responding to the Examiner's prior art rejections, Assignee here only discusses the patentability of the independent claims (*i.e.,* claims 1, 22, 28, 43, 58, 63, 74, 75, 79, 84 and 85). As the Examiner will appreciate, should these independent claims be patentable over the prior art, dependent claims would also necessarily be patentable. Accordingly, Assignee does not separately discuss the patentability of the dependent claims, although Assignee reserves the right to do so. Portions of the office action have been copied into this reply to aid the Examiner.

Section 101 Rejections

The Examiner has rejected claims 79-85 under 35 U.S.C. §101 as being allegedly being directed to non-statutory subject matter. Specifically the Examiner has rejected these claims as follows:

> 16. Claims 79-83 are directed to image processing application program interface embodied on one or more computer readable media. Specification does not appear to explicitly define "computer readable media". Specification defines application program interface as a high-level programming language (p. 9, [0047]). Since specification does not define "computer readable media", one of ordinary skill in the art would fairly determine image processing application

> program interface (high-level programming language) could be embodied on "computer readable media" that could include signals or other forms of propagation and transmission media in order to transmit image processing application program interface (high-level programming language). Signals or other forms of propagation and transmission media fail to be appropriate manufacture under 35 U.S.C. 101 in context of computer-related inventions, and thus "computer readable media" is directed to non-statutory subject matter (see MPEP 2106 IV).
>
> 17. Claim 84 is directed to application program interface. Specification defines API as high-level programming language (p. 9, [0047]). Functional descriptive material consists of computer programs. Descriptive material is nonstatutory when claimed as descriptive material per se (see MPEP 2106.01). So, API is directed to non-statutory subject matter.
>
> 18. Claim 85 is directed to computer-readable medium having computer executable instructions. Specification does not appear to explicitly define "computer-readable medium". So, one of ordinary skill in the art would fairly determine "computer-readable medium" could include signals or other forms of propagation and transmission media in order tô transmit the instructions. Signals or other forms of propagation and transmission media fail to be appropriate manufacture under 35 U.S.C. 101 in context of computer-related inventions, and thus "computer-readable medium" is directed to non-statutory subject matter (see MPEP 2106 IV).

Office Action dated 30 January 2008 at p. 5-6.

Regarding claims 79-84 and 85 the Examiner asserts the specification does not appear to explicitly define "computer-readable medium" and that one of ordinary skill in the art would fairly determine this to include signals or other forms of propagation and transmission media. The Examiner is directed to the original specification at least at pg. 5 ¶ 32. This portion of the specification states "memory or any other suitable storage medium that exists or may exist in the future." Furthermore, specific examples (*e.g.* RAM, computer disk drive, non-volatile storage, optical memory, USB devices) of storage are listed here and throughout the specification. Assignee points out that one of ordinary skill in the art will recognize that a computer readable medium is clearly defined in the specification. Assignee respectfully requests Examiner withdraw this objection.

Additionally, the Examiner asserts these claims pertain to non-statutory subject matter because it is "descriptive material *per se*" because the Examiner contends that the specification (at p. 9, [0047]) "defines API as high level programming language."

The Examiner has misunderstood the cited section of the specification. The cited section is describing "Core Imaging" and specifically states "Core Imaging is a software suite comprising ... a high level language ***or*** API built for graphics functions." The explicit use of the term ***or*** certainly does not equate an API and a high level programming language as the Examiner contends. Furthermore, one of ordinary skill in the art would clearly understand that an API is an "application program interface" and not equivalent to a high level programming language because it is an ***interface*** and not a language.

Furthermore, each of these claims includes a concrete and tangible step of processing an image and cannot be classified as "nonfunctional" because it recites, among other limitations, functions to "create image objects" and "convert image objects into context objects" which each perform a defined function. Furthermore, "Only when the claimed invention taken as a whole is directed to a mere program listing *i.e.*, to only its description or expression, is it descriptive material *per se* and hence nonstatutory.... When a computer program is claimed in a process where the computer is executing the computer program's instructions, USPTO personnel should treat the claim as a process claim." MPEP § 2106.01. Therefore the requirements of 35 U.S.C. § 101 are clearly met. Assignee respectfully requests Examiner withdraw this rejection.

Section 102 Rejections

The Examiner has rejected claims 28-29, 38-39, 43-44, 53-54 and 74 under 35 U.S.C. § 102(b) as allegedly being anticipated by U.S. Patent No. 6,215,495 to Grantham ("Grantham").

Grantham

Grantham is directed to "a platform independent application program interface for interactive three-dimensional scene management." Grantham at Abstract. Grantham further discloses "[i]n particular, the applications programming interface is used to render a three-dimensional scene according to a download file. This is accomplished by building a scene graph from a number of different objects which are stored in local memory [as opposed to a server on the internet]. These objects have variables which can be changed by subroutine calls." Grantham at Col. 2, lns. 54-59. Grantham discloses a use for Virtual Reality Meta Language (VRML) in recognition of the limited bandwidth of the internet to more efficiently utilize the existing bandwidth in the transmission of 3-D image information. *See* Grantham at Col. 1, lns. 64-67.

To summarize, Grantham discloses a use for the ***interpreted*** VRML language that stores persistent data objects in the computer's local memory such that the server, which is located at some point in the internet, is saved from having to re-transmit this type of data each and every time a new scene graph is to be constructed. *See* Grantham at Col. 4, lns. 50-55. Of particular note, Grantham repeatedly states that VRML is an interpreted language (*See eg.* Col. 5 lns. 5, 8; Fig. 1 element 111) and that the "CompileAction class 403 compiles a specified subgraph into a ***data structure*** which is more efficient for traversals." Grantham at Col. 7, lns. 50-52. The "CompileAction" is NOT "compilation", in its normal defined sense or as used by applicant, because Grantham's "CompileAction" does NOT yield executables.

The Examiner has specifically rejected independent claim 28 as follows:

21. As per Claim 28, Grantham teaches processor 108 initiates request which is routed to server 101. When server 101 receives such request, processor 102 retrieves appropriate VRML file. VRML file instructs API 112 to make a number of function calls to various graphics engines 113 for performing desired functions on persistent data objects 118. API is structured as collection of class hierarchies (c. 4, ll. 63-c. 5, ll. 23), including Graphics State classes that includes Context, Appearance, Material, Texture, and TexTranform classes (c. 28, ll. 14-16). Context class maintains graphics state for particular graphics context (c. 3, ll. 2-4). The other Graphics State classes define how result image is to be created (c. 8, ll. 38-46). DrawAction class is used to draw scene (c. 7, ll. 31-33). So Grantham teaches method of creating result image

> having 1st process requesting creation of context; 1st process requesting creation of result image; 1st process indicating parameters associated with creation of result image; 1st process requesting result image be rendered to context (c. 4, ll. 63-c. 5, ll. 23; c. 28, ll. 14-16; c. 3, ll. 2-24; c. 8, ll. 38-46; c. 7, ll. 31-33); 2nd process servicing requests of 1st process, servicing comprising steps of optimizing graph representing result image; compiling optimized graph; causing rendering of compiled optimized graph into context (c. 5, ll. 2-23; c. 4, ll. 55-58; c. 7, ll. 50-57; c. 3, ll. 2-4).

Office Action dated January 30, 2008 at p. 6-7.

As stated above, Grantham discloses communication between an internet server and a local processor's web browser. The Examiner interprets Grantham such that the 1st process is run on the local machine 108 and the second process is performed by the internet server 101. The Examiner then asserts "1st process indicating parameters associated with creation of result image." This assertion is incorrect because Grantham discloses that an internet request is initiated by the local machine and "[w]hen server 101 receives such a request, processor 102 retrieves the appropriate VRML file from memory 104. The VRML file is then transmitted back over the Internet ... The VRML file instructs the API 112 to make a number of function calls." Grantham at Col. 4, ln 66 through Col. 5., ln 3. Therefore it is clear that the local processor (i.e. 1st process) does not "indicate parameters associated with creation of result image" as recited in the claim. The local processor is merely making an internet request of a web server and the web server provides the appropriate VRML file. The cited "first process" in Grantham merely asks for the VMRL file without any reference to or cognizance of the claimed "creation of context", "creation of result image", or "parameters associated with result image".

Furthermore, Grantham does not disclose the steps of both optimizing the graph and compiling the optimized graph as recited in claim 28. The Examiner cites to the CompileAction class disclosed in Grantham to anticipate this limitation. As stated above, VRML is an ***interpreted*** language and the CompileAction class creates a ***data structure*** which is more efficient for traversals. Even though Grantham has named the example class "CompileAction" the function of the class is to optimize a data structure for traversals and not the standard use of compiling which produces an executable. *See* Specification at ¶ 70. As stated above, Grantham's "CompileAction" does NOT produce an executable and thus, is not "compiling".

Because Grantham does not disclose each and every limitation of claim 28 the Examiner has failed to present a legitimate *prima facie* case of anticipation. Furthermore, Each of claims 29 and 38-39 depend from claim 28 and are not anticipated by Grantham for at least the same reasons. Assignee respectfully requests the Examiner withdraw this rejection.

The Examiner has specifically rejected independent claim 43 as follows:

> 25. As per Claim 43, it is similar in scope to Claim 28, except it has additional limitation that 1st process and 2nd process are running on first microprocessor, and rendering occurs on second microprocessor. Grantham teaches first process (requesting) and second process (compiling) are running on first microprocessor (108), and rendering occurs on second microprocessor (109) (c. 4, ll. 63-c. 5, ll. 23; c. 7, ll. 50-57). So Claim 43 is rejected under same rationale as Claim 28.

The Examiner asserts that the first process is the requesting process disclosed in Grantham. As stated above, the requesting process in Grantham is a web browser and the web browser does not perform the limitations of "requesting the creation of a result image", "indicating parameters associated with the creation of said result image" and "requesting that said result image be rendered to said context" as recited in claim 43. The internet server of Grantham performs the function of creating the VRML file. The arguments stated above with respect to claim 28 also apply to claim 43.

Because Grantham does not disclose each and every limitation of claim 43 the Examiner has failed to present a legitimate *prima facie* case of anticipation. Furthermore, Each of claims 44 and 53-54 depend from claim 43 and are not anticipated by Grantham for at least the same reasons. Assignee respectfully requests the Examiner withdraw this rejection.

The Examiner has specifically rejected independent claim 74 as follows:

> 27. As per Claim 74, Grantham teaches rendering image, 1st process running on CPU 107 requesting creation of image; graphics services resource 109, responding to request by running 1st routine on CPU, 1st routine for optimizing graph representing image; 1st process requesting rendering of image to specified context; graphics services resource causing rendering of graph representing image, rendering occurring on GPU (c. 4, ll. 55-58, 63-67; c. 5, ll. 1-23, 61-67).

The 1st process of Grantham is a web browser and is not "requesting creation of an image" as recited in claim 74. The web browser is "initiating a request which is routed by input/output (I/O) device 107 through the internet 106 to server 101." Grantham at Col. 4, lns. 63-67. A web browser requesting information from a server is not the same as "a first process running on a CPU requesting creation of an image" as recited in claim 74. Furthermore, the web browser does not have "a graphics services resource, responding to said request" because the internet server 101 is what responds to the request disclosed in Grantham.

Because Grantham does not disclose each and every limitation of claim 74 the Examiner has failed to present a legitimate *prima facie* case of anticipation. Assignee respectfully requests the Examiner withdraw this rejection.

The Examiner has rejected claims 75 and 78 under 35 U.S.C. § 102(e) as allegedly being anticipated by U.S. Patent No. 6,995,765 to Boudier ("Boudier").

Boudier

Boudier is directed to optimization of a scene graph via an optimization base that contains a set of specific atomic optimizations, an optimization registry that lists each atomic optimization, parameters associated with each optimization and priority information relating to the necessary order in which optimizations must be performed. Boudier at Abstract. "The selection of a specific atomic optimization is made by user 410. User 410 supplies user configuration information 415 to optimization manager 440 via a configuration manager 420." Boudier at Col. 4, lns. 49-54. To summarize, Boudier discloses a system that takes ***user input*** to perform specific atomic optimization steps for a particular scene graph.

The Examiner has rejected independent claim 75 specifically as follows:

> 30. As per Claim 75, Boudier teaches converting 1st image representation into 2nd image representation, comprising creating 1st graph associated with 1st image representation (c. 1, ll. 29-36; c. 2, ll. 31-34; c. 3, ll. 61-63) where software routines for creating such graph execute on CPU (504) (c. 1, ll. 49-51: c. 6, ll. 34-38), determining intersection of 1st graph's global domain of definition (input scene graph) and global region of interest (bounding box) (c. 9, ll. 12-21); resolving 1st node in 1st graphic by running software routines on CPU to determine if 1st node may be combined with 2nd node (c. 9, ll. 40-53; c. 5, ll. 33-37) and create program steps for calculating and storing only portions of any intermediary image that relate to intersection (c. 9, ll. 12-21), program steps for compilation on CPU, execution on GPU (c. 5, ll. 33-37; c. 6, ll. 19-21).

Office Action dated 30 January 2008 at p. 8.

The Examiner asserts that the 1st graph's global domain of definition is the input scene graph of Boudier. This is simply incorrect. Assignee directs Examiner to the instant specification at ¶ 92 which states "[a] practical way to think about domain of definition is as a representation of all places in an image that are explicitly defined and not transparent." The scene graph of Boudier is a model with the nodes representing features of a scene and edges representing associations between the connected components. *See* Boudier at Col. 1, lns. 30-34.

The Examiner similarly asserts that the global region of interest is comparable to the bounding box of Boudier. Again this assertion is incorrect. A bounding box is defined as the "a cupoid, or in 2-D a rectangle, containing the object." *See* http://en.wikipedia.org/wiki/Bounding_volume. That is the cuboid formed when rectangles are used to outline an objects left, right, front, back, top and bottom. This is in stark contrast to the definition of global region of interest defined in the instant specification at ¶¶ 98 and 100 which explain that an region of interest "is the portion of the input image that is necessary to compute the given DOD."

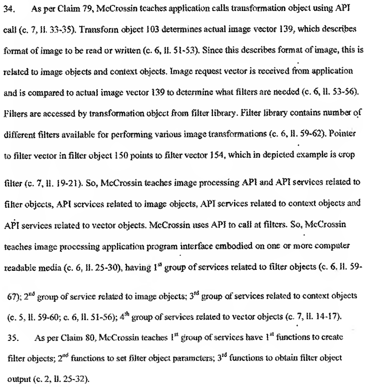
Boudier is silent regarding the region of interest or domain of definition as used in claim 75. Because Boudier does not disclose each and every limitation of claim 75 the Examiner has failed to present a legitimate *prima facie* case of anticipation. Furthermore, claim 78 depends from claim 75 and is not anticipated by Boudier for at least the same reasons. Assignee respectfully requests the Examiner withdraw this rejection.

The Examiner has rejected claims 79-84 under 35 U.S.C. § 102(e) as allegedly being anticipated by U.S. Patent No. 6,600,840 to McCrossin et al. ("McCrossin").

McCrossin

McCrossin is directed to a system for changing the format of an image by applying a series of filters from a filter stack. The filters are applied to the source image for each parameter of the source image that does not match a parameter of the requested or desired result image. In essence, McCrossin proposes a system to simplify the conversion of the same image from one format to another; for example from a BMP file to a TIFF file. See for example Col. 6 line 40 *et. Seq.*

The Examiner has rejected independent claim 79 specifically as follows:



> 34. As per Claim 79, McCrossin teaches application calls transformation object using API call (c. 7, ll. 33-35). Transform object 103 determines actual image vector 139, which describes format of image to be read or written (c. 6, ll. 51-53). Since this describes format of image, this is related to image objects and context objects. Image request vector is received from application and is compared to actual image vector 139 to determine what filters are needed (c. 6, ll. 53-56). Filters are accessed by transformation object from filter library. Filter library contains number of different filters available for performing various image transformations (c. 6, ll. 59-62). Pointer to filter vector in filter object 150 points to filter vector 154, which in depicted example is crop filter (c. 7, ll. 19-21). So, McCrossin teaches image processing API and API services related to filter objects, API services related to image objects, API services related to context objects and API services related to vector objects. McCrossin uses API to call at filters. So, McCrossin teaches image processing application program interface embodied on one or more computer readable media (c. 6, ll. 25-30), having 1st group of services related to filter objects (c. 6. ll. 59-67); 2nd group of service related to image objects; 3rd group of services related to context objects (c. 5, ll. 59-60; c. 6, ll. 51-56); 4th group of services related to vector objects (c. 7, ll. 14-17).
>
> 35. As per Claim 80, McCrossin teaches 1st group of services have 1st functions to create filter objects; 2nd functions to set filter object parameters; 3rd functions to obtain filter object output (c. 2, ll. 25-32).

Office Action dated 30 January 2008 at p. 9.

As discussed earlier, McCrossin proposes a system for changing the format of an image by applying a series of filters. Thus, the API discussed by McCrossin does not disclose an image processing API as set forth in claim 79 rather, McCrossin's only APIs are "read" and "write" APIs associated with, "an improved method and system for transforming image data from one format to another wherein the number of converters required is reduced." McCrossin at Col. 1 lines 19-21. The Examiner cites to, "An application (not shown) calls the transform object using a procedure call or API call, which results in call 166 being made." (McCrossin at Col. 7, lines 33-35). As can be clearly seen in Figures 8A and 8B, element 166 is a common read function call (specifically "fread"). Therefore, McCrossin is not teaching any kind of API here other than a common "read" API that performs a read function on an image whose format will be changed under McCrossin's principles. The Examiner is incorrect in inferring that McCrossin teaches the specific bundle of API services as claimed by merely stating that an API call or procedure call is used. Simply put, the disclosure in McCrossin comprises only read and write APIs and a translator of image formats. In dramatic contrast, Assignee's invention in claim 79 is for a specifically claimed "image processing application program interface;" i.e. APIs to filter objects, APIs to image objects, APIs to context objects and APIs to vector objects. These two concepts are abundantly different.

The Examiner also states that "Transform object 103 determines an actual image vector 139, which describes the format of the image to be read or written" (McCrossin at Col. 6, lines 51-53). The Examiner then states "Since this describes the format of the image, this is related to image objects and context objects." (See Office Action dated 30 March 2007 at ¶ 19 page 18). Again the Examiner has made an incorrect extrapolation, simply describing the format of the image does not encompass the requirements of image objects and context objects. As will be seen in the discussion of claim 84 below, the Examiner appears to have an incorrect understanding of the "context objects" as disclosed and claimed by Assignee. For example, in the Office Action dated 30 March 2007 at ¶ 24 page 19, the Examiner misquotes Assignee's specification to say, "creating a context is derived from tools that allow the definition of an object in memory." The correct quotation from Assignee's specification is "the ***ability*** to create a context is derived from tools that allow definition of an object in memory." This simply means that Assignee is teaching that a context may be created using "tools that allow definition of an object in memory." The Examiner seems to have incorrectly interpreted the partial quotation to believe that any definition of an object in memory is creating a context. This is simply not true and not what is stated in Assignee's claims or disclosure. McCrossin does not teach or discuss context creation and the Examiner's belief to the contrary appears to be based on his belief that any definition of an object in memory is creating a context.

McCrossin does not and can not disclose the bundle of expressly claimed API services; i.e. API services related to filter objects, API services related to image objects, API services related to context objects and API services related to vector objects. Such a bundle of API services as claimed is entirely absent from McCrossin at least because a major purpose of McCrossin is to isolate the application layer from the task of doing a file format conversion. For example, the McCrossin application uses a common "read" or "write" API to obtain a file that is in a format associated with the isolated application program, McCrossin's API does NOT call any or all "filters" involved in the file format conversion). The Examiner has found a single reference to "a procedure call or API call" in McCrossin at Col. 7, lines 33-35 and inappropriately extrapolated this reference to contend that McCrossin teaches the specifically claimed limitations of independent claim 79. Thus, in view of this clarification, Assignee submits that claim 79 should be in condition for allowance. Claims 80 through 83 depend from claim 79 and therefore should be allowable for the same reasons. Assignee respectfully requests the Examiner withdraw this rejection.

The Examiner has rejected independent claim 84 specifically as follows:

> 39. As per Claim 84, McCrossin teaches application calls transformation object using API call (c. 7, ll. 33-35). Filters are accessed by transformation object from filter library 143. Filter library contains number of different filters available for performing various image transformations (c. 6, ll. 59-62). Transform object adds filter to filter stack to create new filter object (c. 6, ll. 51-58; c. 8, ll. 25-37). According to Applicant's disclosure, image, more commonly, may be created from another image by applying filter to existing image ([0057], p. 12). So, McCrossin teaches API functions to create image objects; API functions to create filter objects. Transform object 103 determines actual image vector 139, which describes format of image to be read or written (c. 6, ll. 51-53). According to Applicant's disclosure, creating context is derived from tools that allow definition of object in memory ([0055], p. 11). Since actual image vector describes format of image to be read or written, this allows definition of object in memory, so is considered to create context objects, so McCrossin teaches API functions to create context objects. Pointer to filter environment in filter object 150 points to memory 156, which contains information which may be used by filter vector 154 in converting or transforming image data (c. 7, ll. 21-24), and so McCrossin does teach API functions to set filter object parameters. Filters are employed by transformation object 103 to provide necessary transformation of image data to return image data in form as specified in image request vector 127 (c. 6, ll. 66-c. 7, ll. 2), and so McCrossin teaches API functions to obtain filter object output; API functions to convert image objects to context objects. So, McCrossin teaches application program interface for facilitating image processing (c. 6, ll. 25-30), API having functions to create image objects; create context objects (c. 6, ll. 51-53); create filter objects; set filter object parameters; obtain filter object output (c. 2, ll. 25-32); convert image objects into context objects (c. 6, ll. 51-53).

Office Action dated 30 January 2008 at p. 10-11.

Similar to claim 79, the Examiner asserts that claim 84 is invalid over McCrossin. Assignee submits that claim 84 is in condition for allowance because, as explained with respect to claim 79, McCrossin discloses only read and write APIs in the context of image format translation and does not and can not disclose, the specifically claimed a bundle of API functions; i.e. APIs to create image objects; APIs to create context objects; APIs to create filter objects; APIs to set filter object parameters; APIs to obtain filter object output; and APIs to convert image objects to context objects. In particular the Examiner cites McCrossin at Col. 6 lines 59-62, which clearly states that "Filters are accessed ... from filter library". The Examiner has found a reference in McCrossin that discloses the application of a filter to an image. However, it is not valid for the Examiner to extrapolate that reference to an API to create filter objects rather than retrieve them from a pre-existing library. In response to previous arguments the Examiner asserts the following:

> In reply, McCrossin teaches application calls transformation object using API call (c. 7, ll. 33-35), transform object adds filter to filter stack to create new filter object (c. 6, ll. 51-58; c. 8, ll. 25-37).

Office Action dated 30 January 2008 at p. 4.

The Examiner has incorrectly paraphrased the citation because there is no teaching in the cited disclosure related to "create new filter object". The cited section of McCrossin does not create a new filter object but creates a "new file object" and filters are only pushed and popped from a ***filter stack***. *See for example* Figs. 6 and 7 which depict a representation of the filter stack. Adding pre-existing filters selected from a filter library to a stack such that they can be applied in a particular order cannot be interpreted as creating new filters.

As explained above for claim 79, the Examiner seems to have incorrectly interpreted the partial quotation to believe that any definition of an object in memory is creating a context. This is simply not true and not what is stated in Assignee's disclosure. McCrossin does not teach or discuss context creation and the Examiner's belief to the contrary appears to be based on his belief that any definition of an object in memory is creating a context. Thus, in view of this clarification, Assignee submits that claim 84 should be in condition for allowance. Assignee respectfully requests the Examiner withdraw this rejection.

Section 103 Rejections

The Examiner has rejected claims 1-2, 8, 11-12, 14-20, 36-37, 40-41, 51-52, 55-56, 58, 61-63, 66-73 and 85 under 35 U.S.C. § 103(a) as allegedly being obvious over Grantham in view of McCrossin.

The Examiner has specifically rejected independent claim 1 as follows:

> 44. As per Claim 1, Grantham teaches processor 108 initiates request which is routed to server 101. When server 101 receives such request, processor 102 retrieves appropriate VRML file. VRML file instructs API 112 to make a number of function calls to various graphics engines 113 for performing desired functions on persistent data objects 118. Interpreter 111 modifies scene graph to suit specific graphics subsystem hardware 109. Hence, interpreter 111 enables VRML file to be adapted to run on graphics subsystem hardware (c. 4, ll. 63-c. 5, ll. 11). API is structured as collection of class hierarchies. Nodes and graphics states are assembled into cohesive scene graph (c. 5, ll. 18-23). Graphics State classes include Texture class 504 that defines how image is filtered (c. 28, ll. 10-16; c. 11, ll. 22-25). Outside Scene Graph classes include CompileAction class 403 that compiles specified subgraph into data structure which is more efficient for traversals (c. 28, ll. 25-27, 29-30; c. 7, ll. 50-57). So filter is part of scene graph (c. 5, ll. 18-23; c. 28, ll. 10-16; c. 11, ll. 22-25) that is run on graphics subsystem hardware (c. 5, ll. 5-11). Before scene graph is run on graphics subsystem hardware (c. 5, ll. 5-11), CompileAction class compiles it into data structure which is more efficient for traversals (c. 7, ll. 50-57), so compiled data structure includes filter. So VRML file requested by processor 108 (c. 4, ll. 63-c. 5, ll. 11) includes filter (c. 5, ll. 18-23; c. 28, ll. 10-16; c. 11, ll. 22-25), and processor 108 requests filter from compiling process since program needs to be compiled before it is run on graphics system hardware. So Grantham teaches method of editing initial image, having steps of 1st process requesting (c. 4, ll. 63-c. 5, ll. 11) filter from 2nd process; related filter and initial image comprising program, second process compiling program, yielding compiled program; running at least portion of compiled program to apply function of filter to image (c. 5, ll. 5-11, 18-23; c. 28, ll. 10-16; c. 11, ll. 22-25; c. 7, ll. 50-57), yielding pixel-image result (c. 5, ll. 9-11).
>
> However, Grantham does not teach 1st process defines relationship between filter and initial image. But, McCrossin teaches editing initial image, 1st process requesting filter from 2nd process; 1st process defining relationship between filter and initial image (c. 6, ll. 46-c. 7, ll. 9).
>
> It would have been obvious to one of ordinary skill in the art at the time of invention by applicant to modify Grantham so first process defines relationship between filter and initial image because McCrossin suggests reducing amount of processing required to convert image data from original or input format into desired output format (c. 1, ll. 54-57; c. 2, ll. 3-24).

Office Action dated 30 January 2008 at p. 12-13.

As stated above the CompileAction class of Grantham does not compile in the same manner as Assignee's claim because it merely produces a data structure which is optimized for traversal and not a compiled program as recited in claim 1. Additionally, CompileAction is a class and not a separate process as recited in claim 1.

Furthermore, the Examiner relies on McCrossin to disclose multiple processes and as stated above, McCrossin simply translates images from one format to another and is silent regarding multiple processes and a specific division of work between the multiple processes as recited in claim 1. Thus, in view of this clarification, Assignee submits that claim 1 should be in condition for allowance. Assignee respectfully requests the Examiner withdraw this rejection.

The Examiner has specifically rejected independent claim 58 as follows:

> 59. As per Claim 58. Grantham teaches interpreter 111 enables VRML file to be adapted to run on graphics system hardware 109 (c. 5, ll. 5-9). VRML is high level programming language (c. 1, ll 63-67). So Grantham teaches providing high level interface (111) to graphics processing resource (109) (c. 5, ll. 5-9; c. 1, ll. 63-67) having 1st process requesting performance of task through 1 or more function calls serviced by graphics processing resource (c. 4, ll. 63-c. 5, ll. 23), function calls selected from 1 or more of following options, (i) creating of image (Graphics State classes) (c. 28, ll. 14-16; c. 8, ll. 37-46; c. 7, ll. 31-33), (iv) asking for output of filter or
>
> another filter (504; c. 11, ll. 22-25), (v) creating context (302); (vi) rendering image to context or another context; request having object associated therewith, object comprising one of following: context (c. 5, ll. 61-67), image (c. 28, ll. 14-16; c. 8, ll. 37-46; c. 7, ll. 31-33), filter (c. 11, ll. 22-25), or vector (c. 26, ll. 20-22); graphics processing resource servicing request (c. 5, ll. 2-23).
>
> However, Grantham does not explicitly teach request defines relationship between at least one of functions and one of objects. However, McCrossin teaches this limitation (c. 6, ll. 46-c. 7, ll. 9). This would be obvious for the same reasons given in the rejection for Claim 1.

Office Action dated 30 January 2008 at p. 15-16.

The Examiner has not provided a reference for either limitation of "(ii) creating a filter" or "(iii) setting arguments associated with said filter" and both McCrossin and Grantham are silent as to creating a filter. As stated above, the filter stack in McCrossin can not be properly interpreted as creating a filter because it merely applies pre-defined filters in a particular order. Thus, in view of this clarification, Assignee submits that claim 58 should be in condition for allowance. Assignee respectfully requests the Examiner withdraw this rejection.

The Examiner has specifically rejected independent claim 63 for the same rationale as claim 58 discussed above. The arguments regarding claim 58 apply with equal force to claim 63. Thus, in view of this clarification, Assignee submits that claim 63 should be in condition for allowance. Assignee respectfully requests the Examiner withdraw this rejection.

The Examiner has specifically rejected independent claim 85 as follows:

69. As per Claim 85, Grantham teaches computer executable instructions for performing method recited in Claim 1 (c. 4, ll. 63-c.5, ll. 23).

Office Action dated 30 January 2008 at p. 17.

The arguments regarding claim 1 apply with equal force to independent claim 85. Thus, in view of this clarification, Assignee submits that claim 85 should be in condition for allowance. Assignee respectfully requests the Examiner withdraw this rejection.

The Examiner has rejected claims 22-23 and 25-27 under U.S.C. § 103(a) as allegedly being obvious over Grantham and McCrossin further in view of U.S. Patent No. 6,600,840 to Hoppe ("Hoppe").

The Examiner has specifically rejected independent claim 22 as follows:

84. As per Claim 22, Grantham teaches system for editing initial image, comprising first microprocessor (108, 111, 112, Fig. 1) running first process and second process (compiling) for servicing requests from first process (c. 4, ll. 63-c. 5, ll. 11; c. 7, ll. 50-57). API is structured as a collection of class hierarchies (c. 5, ll. 16-23), and these classes include Texture class 504 that defines how image is filtered (c. 11, ll. 22-25). So filter is in API (c. 5, ll. 16-23; c. 11, ll. 22-25), and API 112 is stored in memory 110, as shown in Fig. 1. So memory 110 is for storing filter, second memory for storing data structure comprising information used in first process (c. 5, ll. 16-23; c. 11, ll. 22-25; Fig. 1) (first and second memories are the same, as recited in Claim 23); second microprocessor (109) for running program created using data structure; outputting pixel image resulting from running program (c. 5, ll. 2-11).

But, Grantham doesn't teach data structure has relationship between initial image and filter. But, McCrossin teaches data structure 103 has relationship between initial image and specific filter (*image request vector 127 is received from application 101 and is compared to actual image vector to determine what filters are needed*, c. 6, ll. 46-c. 7, ll. 9; *application calls the transformation object using API call*, c. 7, ll. 33-35). This is obvious for reasons for Claim 1.

However, Grantham and McCrossin do not explicitly teach third memory for storing pixel image resulting from running program. However, Hoppe teaches API (c. 2, ll. 20-26) for filtering (c. 1. ll. 34-36), and third memory (208) for storing pixel image resulting from running program (c. 4, ll. 36-37; c. 9, ll. 33-36; c. 1, ll. 55-58).

It would have been obvious to one of ordinary skill in the art at the time of invention by applicant to modify Grantham and McCrossin to include 3rd memory for storing pixel image resulting from running program because Hoppe suggests it is well-known in the art to have frame buffer in video memory to store rendered image so that rendered image can be output from video memory to display (c. 4, ll. 36-37; c. 9, ll. 33-36).

Office Action dated 30 January 2008 at p. 21-22.

The Examiner asserts that Grantham teaches "a second process (compiling)." The CompileAction class disclosed in Grantham is simply a class for object oriented programming. Grantham is silent as to multiple processes running on a first microprocessor because it is directed to rendering a 3D image in across the internet inside of a web browser.

The Examiner also asserts in rejecting claim 22 that McCrossin teaches data structure 103 has relationship. In response to previous arguments the Examiner states:

In reply, Examiner points out McCrossin teaches transform object 103 has relationship between initial image and specific filter (c. 6, ll. 51-56), so transform object 103 is data structure.

Office Action dated 30 January 2008 at p. 3.

Throughout the office action the transform object 103 has been interpreted as an API or subroutine. See for example rejections to claim 79 and 84. In rejecting claim 22 the Examiner asserts that transform object 103 defines a relationship and anticipates the data structure of independent claim 22. This is simply an improper interpretation of McCrossin. McCrossin does not disclose a data structure comprising a relationship between an image and a filter via transform object 103.

Thus, in view of this clarification, Assignee submits that claim 22 should be in condition for allowance. Furthermore, claims 23 and 25-27 depend from claim 22 and should be patentable for at least the same reasons. Assignee respectfully requests the Examiner withdraw this rejection.

The Examiner has rejected claims 3, 4-7, 9-10, 13, 21, 24, 30-35, 42, 45-50, 57, 59-60, 64-65 and 76-77 under U.S.C. § 103(a) as allegedly being obvious over Grantham and various other references. Each of these claims is a dependent claim and is patentable at least for the reasons already stated regarding its' parent independent claim. Assignee respectfully requests the Examiner withdraw this rejection.

CONCLUSIONS

This paper is intended to be a complete response to the above-identified Office Action. Assignee believes no fees are due. However if it is found that additional fees are due the Commissioner is authorized to deduct the necessary charges from Deposit Account: 501922/119-0033US.

Reconsideration of pending claims 1-85 in light of the above remarks is respectfully requested. If, after considering this Reply, the Examiner believes that a telephone conference would be beneficial towards advancing this case to allowance, the Examiner is strongly encouraged to contact the undersigned attorney at the number listed.

/William M. Hubbard, J.D./
Reg. No. 58,935
Wong, Cabello, Lutsch, Rutherford & Brucculeri, L.L.P.
Customer No. 29855 Voice: 832-446-2445
20333 SH 249, Suite 600 Mobile: 713-302-4648
Houston, Texas 77070 Facsimile: 832-446-2424
Email: whubbard@counselip.com